What is claimed is:

1. A method for operating a computer using object-based computer code, the method comprising:

invoking an event handler method by calling another method of an instance of a class for which parameters passed to the other method are also passed to the event handler method, a parameter list of the other method having a same signature as a parameter list of the event handler method, wherein the other method references the event handler method;

creating an invocation list associated with the other method, the invocation list specifying one or more event handler methods to be invoked; and

dynamically altering contents of the invocation list.

2. The method of claim 1, further comprising adding an event handler method to the invocation list during execution of the object-based computer code.

3. The method of claim 2, further comprising using an event accessor to add the event handler method to the invocation list during execution of the object-based computer code.

4. The method of claim 2, further comprising using an addition operator to add the event handler method to the invocation list during execution of the object-based computer code.

5. The method of claim 1, further comprising removing an event handler method from the invocation list during execution of the object-based computer code.

6. The method of claim 5, further comprising using an event accessor to remove the event handler method from the invocation list during execution of the object-based computer code.

7. The method of claim 5, further comprising using a subtraction operator to remove the event handler method from the invocation list during execution of the object-based computer code.

8. The method of claim 1, wherein the event handler method is configured to handle a static event.

9. The method of claim 8, further comprising referencing the static event using a member access of the form *E.M*, wherein *E* denotes a type and *M* denotes the static event.

10. The method of claim 1, wherein the event handler method is configured to handle a virtual event.

11. The method of claim 10, further comprising overriding the virtual event.

12. The method of claim 1, wherein the event handler method is configured to handle an override event having a signature.

13. The method of claim 12, further comprising identifying an accessible event having a same signature as the override event.

14. The method of claim 1, wherein the event handler method is configured to handle an abstract event.

15. The method of claim 1, wherein the event handler is configured to handle an interface event.

16. The method of claim 1, wherein the parameters passed to the other method and to the event handler method comprise:

a sender parameter identifying an event source; and

an event arguments parameter identifying one or more event arguments.

17. A method for operating a computer using object-based computer code, the method comprising:

invoking an event handler method by calling another method, wherein the other method references the event handler method;

passing a sender parameter identifying an event source and an event arguments parameter identifying one or more event arguments to the other method and to the event handler method, parameter lists of the event handler method and the other method having compatible signatures;

creating an invocation list associated with the other method, the invocation list specifying one or more event handler methods to be invoked; and

altering contents of the invocation list during execution of the object-based computer code.

18. The method of claim 17, further comprising adding an event handler method to the invocation list during execution of the object-based computer code.

19. The method of claim 18, further comprising using an event accessor to add the event handler method to the invocation list during execution of the object-based computer code.

20. The method of claim 18, further comprising using an addition operator to add the event handler method to the invocation list during execution of the object-based computer code.

21. The method of claim 17, further comprising removing an event handler method from the invocation list during execution of the object-based computer code.

22. The method of claim 21, further comprising using an event accessor to remove the event handler method from the invocation list during execution of the object-based computer code.

23. The method of claim 21, further comprising using a subtraction operator to remove the event handler method from the invocation list during execution of the object-based computer code.

24. A computer-readable medium having stored thereon a plurality of computer-executable modules, the computer-executable modules comprising:

an event source module configured to issue an event; and

an event handler module configured to

invoke an event handler method by calling another method of an instance of a class for which parameters passed to the other method are also passed to the event handler method, a parameter list of the other method having a same signature as a parameter list of the event handler method, wherein the other method references the event handler method,

create an invocation list associated with the other method, the invocation list specifying one or more event handler methods to be invoked, and

dynamically alter contents of the invocation list.

25. The computer-readable medium of claim 24, wherein the event handler module is further configured to add an event handler method to the invocation list during execution of the object-based computer code.

26. The computer-readable medium of claim 25, wherein the event handler module is further configured to use an event accessor to add the event handler method to the invocation list during execution of the object-based computer code.

27. The computer-readable medium of claim 25, wherein the event handler module is further configured to use an addition operator to add the event handler method to the invocation list during execution of the object-based computer code.

28. The computer-readable medium of claim 24, wherein the event handler module is further configured to remove an event handler method from the invocation list during execution of the object-based computer code.

29. The computer-readable medium of claim 28, wherein the event handler module is further configured to use an event accessor to remove the event handler method from the invocation list during execution of the object-based computer code.

30. The computer-readable medium of claim 28, wherein the event handler module is further configured to use a subtraction operator to remove the event handler method from the invocation list during execution of the object-based computer code.

31. The computer-readable medium of claim 24, wherein the event source module is configured to issue a static event.

32. The computer-readable medium of claim 31, wherein the event handler module is further configured to reference the static event using a member access of the form *E.M*, wherein *E* denotes a type and *M* denotes the static event.

33. The computer-readable medium of claim 24, wherein the event source module is configured to issue a virtual event.

34. The computer-readable medium of claim 33, wherein the event handler module is further configured to override the virtual event.

35. The computer-readable medium of claim 24, wherein the event source module is configured to issue an override event having a signature.

36. The computer-readable medium of claim 35, wherein the event handler module is further configured to identify an accessible event having a same signature as the override event.

37. The computer-readable medium of claim 24, wherein the event source module is configured to issue an abstract event.

38. The computer-readable medium of claim 24, wherein the event source module is configured to issue an interface event.

39. The computer-readable medium of claim 24, wherein the parameters passed to the other method and to the event handler method comprise:

a sender parameter identifying an event source; and

an event arguments parameter identifying one or more event arguments.

40. A computer-readable medium having stored thereon a plurality of computer-executable modules, the computer-executable modules comprising:

an event source module configured to issue an event; and

an event handler module configured to

invoke an event handler method by calling another method, wherein the other method references the event handler method,

pass a sender parameter identifying an event source and an event arguments parameter identifying one or more event arguments to the other method and to the event handler method, parameter lists of the event handler method and the other method having compatible signatures,

create an invocation list associated with the other method, the invocation list specifying one or more event handler methods to be invoked, and

alter contents of the invocation list during execution of the object-based computer code.

41. The computer-readable medium of claim 40, wherein the event handler module is further configured to add an event handler method to the invocation list during execution of the object-based computer code.

42. The computer-readable medium of claim 41, wherein the event handler module is further configured to use an event accessor to add the event handler method to the invocation list during execution of the object-based computer code.

43. The computer-readable medium of claim 41, wherein the event handler module is further configured to use an addition operator to add the event handler method to the invocation list during execution of the object-based computer code.

44. The computer-readable medium of claim 40, wherein the event handler module is further configured to remove an event handler method from the invocation list during execution of the object-based computer code.

45. The computer-readable medium of claim 44, wherein the event handler module is further configured to use an event accessor to remove the event handler method from the invocation list during execution of the object-based computer code.

46. The computer-readable medium of claim 44, wherein the event handler module is further configured to use a subtraction operator to remove the event handler method from the invocation list during execution of the object-based computer code.

47. The computer-readable medium of claim 40, wherein the event source module is configured to issue a static event.

48. The computer-readable medium of claim 47, wherein the event handler module is further configured to reference the static event using a member access of the form *E.M*, wherein *E* denotes a type and *M* denotes the static event.

49. The computer-readable medium of claim 40, wherein the event source module is configured to issue a virtual event.

50. The computer-readable medium of claim 49, wherein the event handler module is further configured to override the virtual event.

51. The computer-readable medium of claim 40, wherein the event source module is configured to issue an override event having a signature.

52. The computer-readable medium of claim 51, wherein the event handler module is further configured to identify an accessible event having a same signature as the override event.

53. The computer-readable medium of claim 40, wherein the event source module is configured to issue an abstract event.

54. The computer-readable medium of claim 40, wherein the event source module is configured to issue an interface event.

55. The computer-readable medium of claim 40, wherein the parameters passed to the other method and to the event handler method comprise:

a sender parameter identifying an event source; and

an event arguments parameter identifying one or more event arguments.

56. An object-based programming system comprising a computer configured to:

invoke an event handler method by calling another method, wherein the other method references the event handler method;

pass a sender parameter identifying an event source and an event arguments parameter identifying one or more event arguments to the other method and to the event

handler method, parameter lists of the event handler method and the other method having compatible signatures;

create an invocation list associated with the other method, the invocation list specifying one or more event handler methods to be invoked; and

alter contents of the invocation list during execution of the object-based computer code.

57. The object-based programming system of claim 56, wherein the computer is further configured to add an event handler method to the invocation list during execution of the object-based computer code.

58. The object-based programming system of claim 57, wherein the computer is further configured to use an event accessor to add the event handler method to the invocation list during execution of the object-based computer code.

59. The object-based programming system of claim 57, wherein the computer is further configured to use an addition operator to add the event handler method to the invocation list during execution of the object-based computer code.

60. The object-based programming system of claim 56, wherein the computer is further configured to remove an event handler method from the invocation list during execution of the object-based computer code.

61. The object-based programming system of claim 60, wherein the computer is further configured to use an event accessor to remove the event handler method from the invocation list during execution of the object-based computer code.

62. The object-based programming system of claim 60, wherein the computer is further configured to use a subtraction operator to remove the event handler method from the invocation list during execution of the object-based computer code.

63. The object-based programming system of claim 56, wherein the event handler method is configured to handle a static event.

64. The object-based programming system of claim 63, wherein the computer is further configured to reference the static event using a member access of the form *E.M*, wherein *E* denotes a type and *M* denotes the static event.

65. The object-based programming system of claim 56, wherein the event handler method is configured to handle a virtual event.

66. The object-based programming system of claim 65, wherein the computer is further configured to override the virtual event.

67. The object-based programming system of claim 56, wherein the event handler method is configured to handle an override event having a signature.

68. The object-based programming system of claim 67, wherein the computer is further configured to identify an accessible event having a same signature as the override event.

69. The object-based programming system of claim 56, wherein the event handler method is configured to handle an abstract event.

70. The object-based programming system of claim 56, wherein the event handler method is configured to handle an interface event.

71. The object-based programming system of claim 56, wherein the parameters passed to the other method and to the event handler method comprise:

a sender parameter identifying an event source; and

an event arguments parameter identifying one or more event arguments.

72. An object-based programming system comprising a computer configured to:

invoke an event handler method by calling another method of an instance of a class for which parameters passed to the other method are also passed to the event handler method, a parameter list of the other method having a same signature as a parameter list of the event handler method, wherein the other method references the event handler method;

create an invocation list associated with the other method, the invocation list specifying one or more event handler methods to be invoked; and

dynamically alter contents of the invocation list.

73. The object-based programming system of claim 72, wherein the computer is further configured to add an event handler method to the invocation list during execution of the object-based computer code.

74. The object-based programming system of claim 73, wherein the computer is further configured to use an event accessor to add the event handler method to the invocation list during execution of the object-based computer code.

75. The object-based programming system of claim 72, wherein the computer is further configured to use an addition operator to add the event handler method to the invocation list during execution of the object-based computer code.

76. The object-based programming system of claim 72, wherein the computer is further configured to remove an event handler method from the invocation list during execution of the object-based computer code.

77. The object-based programming system of claim 76, wherein the computer is further configured to use an event accessor to remove the event handler method from the invocation list during execution of the object-based computer code.

78. The object-based programming system of claim 76, wherein the computer is further configured to use a subtraction operator to remove the event handler method from the invocation list during execution of the object-based computer code.

79. The object-based programming system of claim 72, wherein the event handler method is configured to handle a static event.

80. The object-based programming system of claim 79, wherein the computer is further configured to reference the static event using a member access of the form *E.M*, wherein *E* denotes a type and *M* denotes the static event.

81. The object-based programming system of claim 72, wherein the event handler method is configured to handle a virtual event.

82. The object-based programming system of claim 81, wherein the computer is further configured to override the virtual event.

83. The object-based programming system of claim 72, wherein the event handler method is configured to handle an override event having a signature.

84. The object-based programming system of claim 83, wherein the computer is further configured to identify an accessible event having a same signature as the override event.

85. The object-based programming system of claim 72, wherein the event handler method is configured to handle an abstract event.

86. The object-based programming system of claim 72, wherein the event handler method is configured to handle an interface event.

87. The object-based programming system of claim 72, wherein the parameters passed to the other method and to the event handler method comprise:

a sender parameter identifying an event source; and

an event arguments parameter identifying one or more event arguments.